EQL-Act: Evidence Query Language Guided Scientific Operation Acting for Scientific VQA

Anonymous ACL submission

Abstract

Scientific Visual Question Answering requires models to interpret quantitative, symbolic, and structural information in scientific figures. Existing reasoning methods improve intermediate reasoning forms, but they often miss task-critical cues or fail to integrate information distributed across regions, modalities, numerical changes, and structural paths. We propose **EQL-Act**, a plug-and-play inference-time framework that casts Scientific VQA as operation-grounded evidence-chain reasoning. EQL-Act routes each question into a task-aware operation space defined by Evidence Query Language (EQL), expands multiple anchor-initiated evidence-query traces, and selects answers by scoring trace portfolios and contrastively verifying trace-level evidence. Experiments on MISS-QA, SpecVQA, and ReactBench show consistent gains across proprietary and open-weight MLLMs, with EQL-Act using GPT-5.1 achieving 75.68% average accuracy and surpassing the strongest non-EQL baseline by 7.57%. Further experiments show that these gains are not simply due to longer reasoning or additional sampling, but arise from the joint effect of task-aware operation routing, anchor-guided trace exploration, and evidence-aware trace verification.

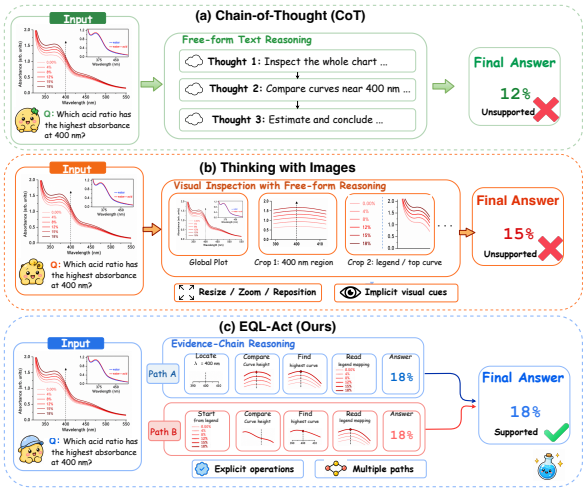

Figure 1: Comparison of Scientific VQA reasoning paradigms. (a) **Chain-of-Thought (CoT)** generates free-form rationales, but the underlying scientific operations remain implicit. (b) **Thinking with Images (TwI)** improves visual inspection through crops, zooms, or repositioning, yet the roles of the obtained visual cues are not explicitly organized. (c) **EQL-Act** represents intermediate reasoning as operation-grounded evidence-query traces and selects the final answer by comparing trace-level support rather than only answer strings.

1 Introduction

Scientific figures communicate experimental observations, quantitative trends, and domain-specific mechanisms (Methani et al., 2020). Unlike general Visual Question Answering (VQA) over natural images, Scientific VQA requires models to interpret quantitative, symbolic, and structural information in scientific representations (Zhang et al., 2023; Zhao et al., 2025). Beyond regular charts with relatively standardized axes, legends, and plotted marks (Masry et al., 2022), broader scientific figures often contain complex diagrams, multi-panel layouts, specialized symbols, and visual-textual relations. Despite recent progress in multimodal large language models (MLLMs), they still struggle with such inputs, especially when answering requires precise value reading, structural understanding, or cross-modal information integration (Zhang et al., 2023; Zhao et al., 2025).

Several paradigms have been explored to improve reasoning in Scientific VQA. Chain-of-Thought (CoT) prompting encourages MLLMs to generate intermediate natural-language rationales before producing the final answer (Lu et al., 2022; Shao et al., 2024). Another line of work, commonly referred to as Thinking with Images, enables models to inspect visual inputs more actively through operations such as cropping, zooming, grounding, or region-level observation (Wu and Xie, 2024;

Qiao et al., 2025). More recently, tool-augmented multimodal methods use external tools or visual programs to assist reasoning, such as invoking detectors, executing visual programs, or calling auxiliary modules during inference (Yang et al., 2023; Gupta and Kembhavi, 2023; Lu et al., 2023). These studies move Scientific VQA beyond one-shot answer generation toward more explicit intermediate reasoning.

However, scientific reasoning is not merely a matter of producing longer rationales or adding more intermediate steps. Existing methods often organize intermediate reasoning as free-form rationales or loosely defined action sequences, such as chain-of-thought explanations or action-based multimodal reasoning trajectories (Lu et al., 2022; Shao et al., 2024; Yang et al., 2023; Gupta and Kembhavi, 2023). These processes can be viewed as largely implicit reasoning procedures, with limited explicit control over how task-relevant information is identified, obtained, and used. As a result, the generated reasoning may appear coherent, yet still overlook necessary information, rely on underspecified visual observations, or fail to connect dispersed information across the figure. Recent Scientific VQA and chart-understanding benchmarks further expose this challenge, where questions may involve cross-panel or multi-chart comparison, fine-grained value reading across regions, symbol-text alignment, or reasoning over diagram structures (Masry et al., 2022; Wang et al., 2024; Borisova et al., 2025). Thus, the key challenge is how to guide models to identify, collect, and integrate the information required by a question across the reasoning process.

Motivated by this challenge, we argue that *reasoning in Scientific VQA should be organized around the progressive acquisition of answer-relevant evidence*. Instead of treating intermediate reasoning as free-form textual expansion, an effective system should clarify what evidence is needed, acquire it through suitable scientific operations, and maintain the acquired information for subsequent reasoning. Moreover, complex scientific questions often require support from different visual regions, modalities, numerical variations, or structural paths, making it insufficient to compress all observations into a single linear rationale. This perspective motivates an operation-grounded evidence reasoning process that preserves how different pieces of evidence are obtained, accumulated, and related to the final answer.

Following this perspective, we propose **EQL-Act**, a plug-and-play inference-time framework for Scientific VQA. Rather than asking an MLLM to directly answer a question or freely generate a rationale, EQL-Act guides the model to first determine what evidence is needed for the question, then acquire the evidence through suitable scientific operations, and finally verify whether the gathered evidence sufficiently supports the answer. For complex questions that require information from different regions, modalities, numerical changes, or structural paths, EQL-Act explores multiple evidence-seeking routes and compares their support before making the final prediction. In this way, EQL-Act turns Scientific VQA from direct answer generation into a structured process of evidence acquisition, accumulation, and verification.

Our contributions are summarized as follows:

- We formulate Scientific VQA as an evidence acquisition problem, in which models must progressively identify, acquire, and verify question-relevant evidence from heterogeneous scientific visuals before producing an answer.
- We propose **EQL-Act**, a plug-and-play inference-time framework with task-aware operation routing, semantic-anchor initiated trace search, and trace-level contrastive selection.
- Extensive experiments on several benchmarks with multiple MLLM backbones demonstrate the performance gains of EQL-Act, while further analyses confirm the effectiveness of its evidence-based reasoning design.

2 Related Work

Scientific Visual Question Answering. Scientific Visual Question Answering requires models to answer questions grounded in scientific visuals, including diagrams, plots, tables, textbook-style figures, and scientific paper figures (Lu et al., 2022; Zhang et al., 2023; Borisova et al., 2025). General vision-language pretraining and multimodal fusion methods provide foundational representations for connecting visual and textual inputs (Kim et al., 2021; Radford et al., 2021), but scientific figures often require more specialized reasoning over numerical values, symbolic notations, structural relations, and domain-specific observations (Kafle et al., 2018; Chaudhry et al., 2020). A large body

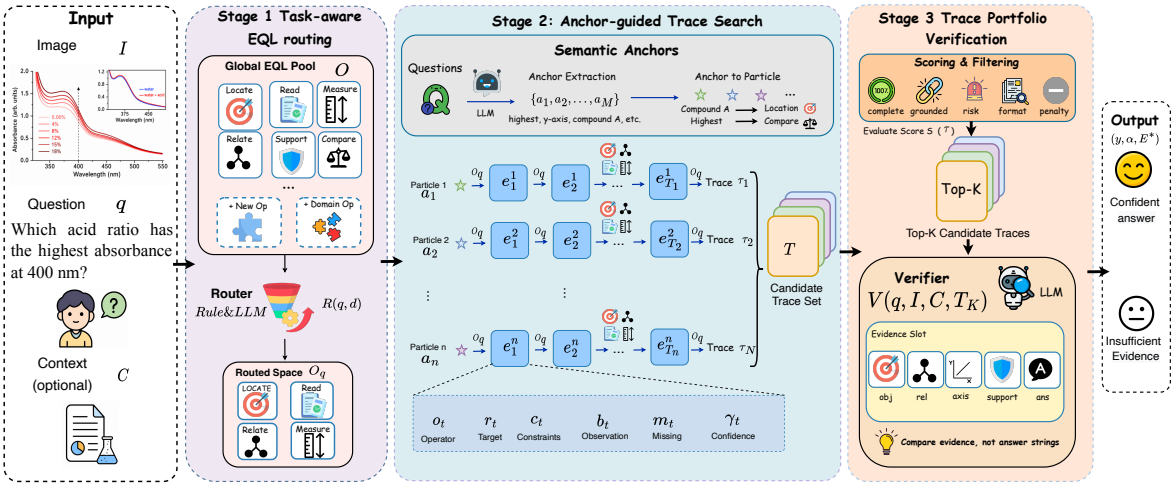

Figure 2: Framework of EQL-Act. Stage 1 routes each question into a task-specific EQL operation space; Stage 2 starts from semantic anchors to search multiple evidence-query traces; Stage 3 scores the trace portfolio and verifies candidate evidence to produce a supported answer or abstain.

of prior benchmarks focuses on chart-like visual structures such as axes, legends, plotted values, and tabular layouts (Methani et al., 2020; Masry et al., 2022; Wang et al., 2024). Recent Scientific VQA benchmarks further broaden the scope to more diverse visual forms, including schematic diagrams, spectral figures, chemical reaction diagrams, experimental figures, and scientific paper contexts (Zhao et al., 2025; Shen et al., 2026; Xu et al., 2026). These settings require models not only to perceive visual elements, but also to identify which information is needed, retrieve or read it from the figure, and connect dispersed observations to support the final answer.

Intermediate Reasoning for Scientific VQA. Recent studies have explored different forms of intermediate reasoning to improve Scientific VQA. CoT-based methods prompt MLLMs to generate natural-language rationales before answering, improving transparency and sometimes reasoning accuracy. ScienceQA provides annotated explanations (Lu et al., 2022), Multimodal-CoT separates rationale generation from answer inference (Zhang et al., 2023), and later works ground or enhance such rationales for visual reasoning (Shao et al., 2024; Movva and Marupaka, 2025). Thinking-with-Images methods encourage models to inspect visual inputs through visual search, grounding, cropping, zooming, or region-level observation. V* performs guided visual search (Wu and Xie, 2024), while V-Thinker supports interactive image inspection (Qiao et al., 2025). Agent methods formulate multimodal reasoning as multi-step action processes, including visual program execution (Gupta and Kembhavi, 2023), Python-based visual inference (Surís et al., 2023), external tool use (Yang et al., 2023), and modular reasoning composition (Lu et al., 2023). Unlike these methods, EQL-Act treats intermediate reasoning as structured evidence-query traces, where operation routing, evidence maintenance, and trace-level support comparison are explicitly controlled.

3 Methodology

3.1 Problem Setting

Given a scientific visual input I, optional context C, and a natural-language question q, Scientific VQA aims to produce an evidence-grounded answer y. The visual input may be a chart, spectrum, schematic diagram, reaction diagram, or multi-panel scientific figure. The context C contains benchmark-provided auxiliary information such as captions, OCR text, retrieved passages, or paper context; otherwise, $C = \emptyset$. Thus, the available evidence consists of I and C.

EQL-Act operates at inference time without updating model parameters. Rather than directly generating y, it constructs candidate evidence-query traces and selects the final answer by trace-level support:

$$(y, \alpha, E^{\star}) = \text{EQL-Act}(I, C, q), \qquad (1)$$

where α denotes whether the answer is sufficiently supported, and $E^{\star}$ is the selected supporting trace.

3.2 Method Overview

As shown in Fig. 2, EQL-Act decomposes inference into three stages. First, *operation-space routing* (§ 3.3) maps the question into a constrained Evidence Query Language (EQL) operation space, specifying admissible operations, active risks, and the expected answer format. Second, *anchor-initiated trace search* (§ 3.4) identifies semantic evidence entry points and expands particles into structured evidence-query traces. Third, *trace-level verification* (§ 3.5) scores, filters, and compares traces to select the final answer. Formally,

$$\begin{aligned} \mathcal{T} &= \mathcal{S}_{\mathrm{EQL}}(I, C, q), \\ \mathcal{T}_K &= \mathcal{K}(\mathcal{T}), \\ (y, \alpha, E^\star) &= \mathcal{V}(I, C, q, \mathcal{T}_K). \end{aligned} \tag{2}$$

Here, $\mathcal{S}_{\mathrm{EQL}}$ denotes EQL-based trace search, including routing, anchor generation, particle initialization, and rollout. $\mathcal{T}$ is the set of generated traces. $\mathcal{K}$ denotes deterministic scoring and Top-K portfolio construction, yielding $\mathcal{T}_K$ for verification. $\mathcal{V}$ is the contrastive verifier, returning answer y, support status α, and selected trace $E^\star$.

3.3 Evidence Query Language and Operation-Space Routing

Evidence Query Language. Inspired by query languages such as SQL, we introduce Evidence Query Language (EQL) as an operation-level protocol for structuring intermediate reasoning. Instead of unconstrained rationales, EQL requires each step to specify the evidence operation, queried target, constraints, and resulting observation:

$$e_t = (o_t, r_t, c_t, b_t, m_t, \gamma_t), \tag{3}$$

where o_t is the operation, r_t the queried target, c_t constraints such as panel, scope, unit, or relation direction, b_t the observation under the target and constraints, m_t missing or uncertain evidence, and γ_t local confidence or support.

The operation o_t must come from a predefined global operation set $\mathcal{O}$. The complete operation pool and field-level query interface are provided in Appendix A.1 and Table 3. Representative operations include `LOCATE`, which restricts evidence search to a panel, region, peak, curve, node, or module, and `COMPARE`, which compares peaks, curves, trends, values, or answer options. EQL is not an external image executor; it is a logical input-output specification imposed on the MLLM during rollout. During trace construction (§ 3.4), the rollout MLLM instantiates EQL steps and records optional candidate answers, support status, and failure modes for later scoring and verification (§ 3.5).

Operation-Space Routing. Given $\mathcal{O}$, EQL-Act derives a compact question-specific operation space before trace search. The router does not inspect the image or predict the answer; it produces

$$R(q, d, \mathcal{O}) = (\mathcal{O}_q, \pi_q, \mathcal{F}_q, \Phi_q), \tag{4}$$

where d is the dataset or domain, $\mathcal{O}_q \subseteq \mathcal{O}$ the routed operation space, π_q an operation prior, $\mathcal{F}_q$ risk flags, and Φ_q the expected answer format.

We implement routing with a Rule + LLM design. Rules map recurring question patterns to conservative operation candidates and risks: panel-related questions suggest `LOCATE`; measurement or unit expressions suggest `CALIBRATE_MEASURE` and `VERIFY_UNIT`; comparison questions suggest `COMPARE`; relation or process questions suggest `INSPECT_RELATION` or `TRACE_PROCESS`; and paper-context questions suggest `RETRIEVE_CONTEXT` and `CHECK_SUPPORT`. The LLM receives the question, dataset identifier, operation descriptions, and rule proposal, and supplements semantically implied operations or risks. The router prompt is provided in Appendix D and Fig. 4. After sanitization, $\mathcal{O}_q$ constrains anchor generation and rollout, π_q guides particle initialization, $\mathcal{F}_q$ guides risk coverage, and Φ_q constrains the answer format.

3.4 Anchor-Initiated EQL Particle Search

Semantic Anchor Generation. The routed operation space $\mathcal{O}_q$ specifies admissible operations, but not where evidence search begins. We define key question cues, such as panels, entities, measurements, relations, or comparison criteria, as *semantic anchors*. An anchor is not a partial answer; it specifies the initial evidence focus of a trace. EQL-Act uses an LLM anchor generator:

$$\mathcal{Z}_q = \{z_1, \ldots, z_M\} = A_{\mathrm{LLM}}(q, C, \mathcal{O}_q, \mathcal{F}_q, \Phi_q). \tag{5}$$

Here, A_{LLM} is the anchor generator, $\mathcal{Z}_q$ the anchor set for q, M the number of anchors, and z_i the i-th anchor. Its inputs are q, C, $\mathcal{O}_q$, $\mathcal{F}_q$, and Φ_q; the prompt template is in Appendix D and Fig. 5. Each z_i contains a question cue, evidence-seeking intent, compatible starting operations $\mathcal{O}_i \subseteq \mathcal{O}_q$, and priority ρ_i. Generated anchors are sanitized by

removing duplicates, clipping priorities, discarding empty fields, ensuring $\mathcal{O}_i \subseteq \mathcal{O}_q$, and excluding RETURN from starting operations.

Anchor-Operation Particle Rollout. EQL-Act initializes particles from anchor-operation pairs. For anchor z_i and compatible operation $o \in \mathcal{O}_i$,

$$p^0_{i,o} = (z_i, o, \tau^0_{i,o}), \qquad \tau^0_{i,o} = \emptyset. \tag{6}$$

Here, $p^0_{i,o}$ is the initial particle started from z_i and o, and $\tau^0_{i,o}$ is its empty trace. A particle is a budgeted evidence-seeking rollout unit, not a free-form rationale sample. If pairs exceed the particle budget B, EQL-Act selects top-ranked particles by

$$s_{\text{start}}(z_i, o) = \rho_i \cdot \pi_q(o), \tag{7}$$

where ρ_i is the anchor priority and $\pi_q(o)$ the router prior for o. This determines the initial evidence focus and first operation.

Each selected particle is expanded step by step. The first step is forced to use o. For later steps, EQL-Act constructs

$$\Gamma(p^t_{i,o}) \subseteq \mathcal{O}_q, \tag{8}$$

where $p^t_{i,o}$ is the particle state after t steps, and $\Gamma(p^t_{i,o})$ the next admissible operation set. It is derived from $\mathcal{O}_q$ according to the current trace, unresolved evidence, $\mathcal{F}_q$, and Φ_q. At each step, the rollout MLLM receives I, C, q, anchor z_i, partial trace $\tau^{<t}_{i,o}$, and $\Gamma(p^t_{i,o})$, then outputs the next EQL step. The rollout prompt is provided in Appendix D and Fig. 6. Rollout stops when RETURN is generated, when a supported answer satisfying Φ_q is produced, or when the step budget is reached. The final trace stores its anchor, operation sequence, candidate answer, support status, confidence, and failure modes.

3.5 Trace Portfolio Scoring and Contrastive Selection

The search stage may produce incomplete, weakly grounded, unsupported, or inconsistent traces. EQL-Act therefore applies deterministic trace scoring before verification. This score is not produced by an additional LLM; it is computed from stored trace fields, including operation validity, observations, missing evidence, candidate answer, support status, confidence, and failure modes, together with $\mathcal{F}_q$ and Φ_q. It filters traces and constructs a compact portfolio, rather than deciding semantic correctness.

Given a completed trace τ, we compute

$$\begin{aligned} S(\tau) = \lambda_c C(\tau) + \lambda_g G(\tau) + \lambda_r R_{\mathcal{F}_q}(\tau) \\ + \lambda_f F_{\Phi_q}(\tau) - \lambda_p P(\tau), \end{aligned} \tag{9}$$

where $C(\tau)$ measures completeness, $G(\tau)$ grounding, $R_{\mathcal{F}_q}(\tau)$ risk coverage, and $F_{\Phi_q}(\tau)$ answer-format compatibility. The penalty $P(\tau)$ aggregates malformed steps, invalid operations, unsupported answers, empty outputs, and failure modes. Detailed scoring design and hyperparameter choices are in Appendix B.

Traces with invalid JSON, illegal operations, invalid support status, transient API errors, or very low scores are filtered out. The remaining traces are sorted by score and confidence:

$$\mathcal{T}_K = \mathrm{TopK}_{\tau \in \mathcal{T}}\big(S(\tau), \mathrm{conf}(\tau)\big), \tag{10}$$

where $\mathcal{T}$ denotes all traces, $\mathcal{T}_K$ the retained Top-K portfolio, and $\mathrm{conf}(\tau)$ is computed from local confidence fields. Sorting uses $S(\tau)$ first and $\mathrm{conf}(\tau)$ as a tie-breaker. We set $K = 3$ in all experiments to retain diverse high-quality traces while limiting verification cost.

The final decision is made by an LLM-based contrastive verifier over $\mathcal{T}_K$. It receives q, I, C, Φ_q, and the Top-K traces; its prompt is in Appendix D and Fig. 7. Unlike rollout, the verifier does not generate new evidence traces. It compares whether traces align with the question focus, address routed risks, support the candidate answer, and leave unresolved missing evidence or failure modes. It then selects one supported candidate answer, optionally normalizes it to Φ_q, or abstains:

$$(y, \alpha, E^\star) = V_{\text{LLM}}(q, I, C, \Phi_q, \mathcal{T}_K), \tag{11}$$

where V_{LLM} is the verifier, y the final answer, α the support status, and $E^\star \in \mathcal{T}_K$ the selected trace. Thus, EQL-Act compares trace-level evidence support rather than voting over answer strings.

4 Experiments

4.1 Experimental Setup

Datasets and Evaluation Metrics. We evaluate EQL-Act on three complementary Scientific VQA benchmarks: MISS-QA for schematic diagram understanding with paper context (Zhao et al., 2025), SpecVQA for spectral understanding and domain-specific reasoning (Shen et al., 2026), and ReactBench for topological reasoning over chemical reaction diagrams (Xu et al., 2026). Detailed

benchmark statistics are provided in Appendix C.1. We follow the official evaluation protocol of each benchmark: ReactBench uses normalized exact match, MISS-QA uses the official LLM-as-judge score in $\{0, 0.5, 1\}$, and SpecVQA uses judge-based accuracy with a 5% tolerance for numeric answers and semantic equivalence for non-numeric answers. All scores are reported as percentages, and Table 1 reports the unweighted macro-average across the three benchmarks.

Baselines. We compare EQL-Act with proprietary MLLMs, open-weight MLLMs, and scientific VQA-related methods. The proprietary baselines include GPT-5.1, GPT-4o, Gemini-2.5-Flash, Claude-4-Sonnet, and Claude-3.5-Sonnet, while the open-weight baselines include Qwen2.5-VL, Qwen3-VL, InternVL2.5, InternVL3.5, Phi-3.5-Vision, DeepSeek-VL2, and LLaVA-Next-Vicuna. We also compare with CoT (Lu et al., 2022), V-Thinker (Qiao et al., 2025), ChartGemma (Masry et al., 2025), Innovator-VL (Wen et al., 2026), E-SVQA (Movva and Marupaka, 2025), and Expert-Neurons (Bhat et al., 2025).

Implementation Details. All methods are evaluated in a zero-shot setting without parameter updates or benchmark-specific fine-tuning. We use a unified inference and evaluation pipeline to ensure consistent image input, optional context formatting, answer normalization, and benchmark-specific scoring. Unless otherwise specified, we set the decoding temperature to 0.2 and the maximum generation length to 1024 tokens. Open-weight models are deployed on NVIDIA A100 80GB GPUs, while proprietary models are accessed through official APIs. For EQL-Act, all internal LLM/MLLM subroutines use the same evaluated backbone unless otherwise stated. We use a hybrid rule-and-LLM router, a particle budget of $B = 6$, a maximum rollout length of 5 EQL steps, and a verification portfolio size of $K = 3$. All intermediate outputs follow predefined JSON schemas and are sanitized before deterministic trace scoring, which does not invoke an additional LLM judge. Detailed scoring rules, hyperparameters, and prompts are provided in Appendices B and D. For methods with rationales or intermediate traces, we evaluate only their final outputs under the official benchmark protocols.

4.2 Main Results

Table 1 reports the main benchmark results across three Scientific VQA datasets and multiple MLLM backbones. Detailed benchmark-specific results are provided in Appendix C.2.

EQL-Act consistently turns strong MLLMs into better scientific reasoners. The strongest variant, EQL-Act w/ GPT-5.1, reaches an average accuracy of 75.68%, outperforming the best non-EQL baseline, Gemini-2.5-Flash, by 7.57 points. The gains are consistent across proprietary backbones: EQL-Act improves GPT-5.1 by 7.97 points, GPT-4o by 9.64 points, and Gemini-2.5-Flash by 6.29 points on average. This indicates that EQL-Act is not tied to a single stronger model, but provides a general inference-time improvement.

The largest gains appear when the task requires structured evidence composition. The improvements are especially pronounced on benchmarks that require relation tracing, scope control, or evidence aggregation. On ReactBench, EQL-Act w/ InternVL3.5 improves the corresponding backbone by 21.05 points; on MISS-QA, EQL-Act improves GPT-4o by 14.64 points and Phi-3.5-Vision by 16.00 points. These results are consistent with our motivation that Scientific VQA benefits from explicit operation routing and trace-level evidence verification.

Structured inference narrows the gap between compact open-weight models and proprietary systems. EQL-Act also brings clear gains to open-weight models. With Qwen3-VL-8B, the average score improves from 55.24% to 63.32%, surpassing Qwen2.5-VL-72B. With InternVL3.5-8B, EQL-Act reaches 73.77% on ReactBench, higher than all non-EQL proprietary baselines on that benchmark.

4.3 Ablation Studies

Table 2 reports core ablations under Qwen3-VL-8B to test whether the gains come from structured evidence-query search rather than isolated prompting or additional sampling. The ablations remove the typed EQL representation, operation-space routing, semantic anchors, stepwise expansion, trace-value estimation, or trace-level verification.

Structured traces, not repeated samples. CoT-SC-6 removes all EQL-specific designs and instead samples six free-form rationales followed by answer voting. Although it uses multiple reasoning attempts, it reaches only 56.44%, 6.88 points below EQL-Act, showing that answer-level diversity alone is insufficient; the key is to construct evidence-bearing traces that can be inspected, compared, and verified.

Routed stepwise search, not an undifferentiated

Model	Params	ReactBench	MISS-QA	SpecVQA	Avg. ↑
Proprietary Multimodal Large Language Models					
GPT-5.1	–	66.74	70.60	65.78	67.71
GPT-4o	–	63.83	63.00	60.78	62.56
Gemini-2.5-Flash	–	64.29	67.30	**72.74**	68.11
Claude-4-Sonnet	–	69.13	71.35	56.88	65.79
Claude-3.5-Sonnet	–	68.60	71.87	55.19	65.22
Open-weight Multimodal Large Language Models					
Qwen2.5-VL	72B	65.18	59.71	58.95	61.28
Qwen3-VL	30B	59.63	57.09	57.36	58.03
Qwen3-VL	8B	58.74	55.40	51.58	55.24
InternVL2.5	38B	48.21	51.60	56.97	52.26
InternVL3.5	8B	52.72	45.80	54.25	50.92
Phi-3.5-Vision	4.2B	39.75	30.30	34.11	34.72
DeepSeek-VL2	16B	36.05	29.86	37.53	34.48
LLaVA-Next-Vicuna	13B	20.73	23.62	29.81	24.72
Scientific VQA-related Methods					
CoT w/ Qwen3-VL	8B	60.75	56.01	53.37	56.71
V-Thinker w/ Qwen2.5-VL	7B	61.24	56.83	52.74	56.94
ChartGemma	7B	65.09	58.94	60.02	61.35
Innovator-VL	8B	62.87	58.21	54.42	58.50
E-SVQA w/ Qwen3-VL	8B	59.61	58.40	56.33	58.11
ExpertNeurons w/ InternVL3	14B	63.22	55.49	62.32	60.34
Our EQL-Act					
EQL-Act w/ GPT-5.1	–	**74.12** ↑7.38	**81.10** ↑10.50	71.81 ↑6.03	**75.68** ↑7.97
EQL-Act w/ GPT-4o	–	71.13 ↑7.30	**77.64** ↑14.64	67.83 ↑7.05	72.20 ↑9.64
EQL-Act w/ Gemini-2.5-Flash	–	**74.75** ↑10.46	73.80 ↑6.50	**74.65** ↑1.91	**74.40** ↑6.29
EQL-Act w/ Qwen3-VL	8B	66.69 ↑7.95	63.70 ↑8.30	59.56 ↑7.98	63.32 ↑8.08
EQL-Act w/ InternVL3.5	8B	73.77 ↑21.05	51.90 ↑6.10	61.36 ↑7.11	62.34 ↑11.42
EQL-Act w/ Phi-3.5-Vision	4.2B	43.29 ↑3.54	46.30 ↑16.00	39.23 ↑5.12	42.94 ↑8.22

Table 1: **Main results on scientific visual question answering benchmarks.** **Red**: best, **Blue**: second best. All results are reported in accuracy (%). ReactBench, MISS-QA, and SpecVQA evaluate reaction reasoning, schematic-diagram information seeking, and scientific plot/spectrum understanding, respectively. Avg. denotes the average score across the three benchmarks.

reasoning space. We next ablate the components that determine how EQL traces are constructed. Among trace-construction variants, Full Op Pool and One-Pass Trace drop to 59.05% and 59.33%, respectively, indicating that both task-specific routing and stepwise expansion are important. No Anchors remains the strongest ablation at 62.81%, but still falls below EQL-Act, suggesting that semantic anchors help initialize complementary evidence-entry points.

Evidence-aware selection, not answer-level aggregation. The final two variants isolate the selection stage after trace construction. No Value removes trace-value estimation before verification, reducing the average score to 60.19%, which shows that weak or incomplete traces should be filtered before the final decision. Answer-only Verifier keeps candidate answers but hides their supporting traces from the verifier, obtaining 60.40%, 2.92 percentage points below EQL-Act. This confirms that the verifier benefits from comparing trace-level evidence rather than only answer strings.

4.4 Further Analysis

To further examine whether the additional inference budget of EQL-Act is effectively converted into performance gains, we conduct a cost-effectiveness analysis against repeated rationale sampling, as shown in Figure 3.

The additional inference budget of EQL-Act is consistently converted into accuracy gains. EQL-Act introduces extra inference cost because it expands multiple evidence-query traces before final verification. As shown in Figure 3(a)–(c), this additional budget leads to substantial macro accuracy gains across backbones, including +11.4 percentage points for InternVL3.5, +9.6 for GPT-4o, +8.1 for Qwen3-VL, and +8.0 for GPT-5.1. These gains form a stronger cost-performance frontier than repeated rationale sampling. Figure 3(d)–(f) further shows that EQL-Act achieves higher marginal utility, such as 2.13 points per API call for InternVL3.5 and 1.92 points per API call for GPT-4o.

Variant	Design Choices						Performance			
	EQL	Router	Anch.	Step	Value	Verifier	ReactBench	MISS-QA	SpecVQA	Avg. ↑
CoT-SC-6	✗	✗	✗	✗	✗	Vote	57.92	59.10	52.31	56.44
Full Op Pool	✓	✗	✓	✓	✓	Trace	63.10	59.40	54.66	59.05
No Anchors	✓	✓	✗	✓	✓	Trace	66.33	63.17	58.94	62.81
One-Pass Trace	✓	✓	✓	✗	✓	Trace	61.16	60.73	56.10	59.33
No Value	✓	✓	✓	✓	✗	Trace	65.61	58.80	56.17	60.19
Answer-only Verifier	✓	✓	✓	✓	✓	Answer	64.77	60.10	56.32	60.40
EQL-Act	✓	✓	✓	✓	✓	**Trace**	**66.69**	**63.70**	**59.56**	**63.32**

Table 2: **Core ablations of EQL-Act under a fixed backbone (Qwen3-VL 8B).** We test whether the gains come from structured EQL traces, operation-space routing, anchor-guided stepwise search, trace-value estimation, and trace-level verification. Avg. denotes the average accuracy across the three benchmarks.

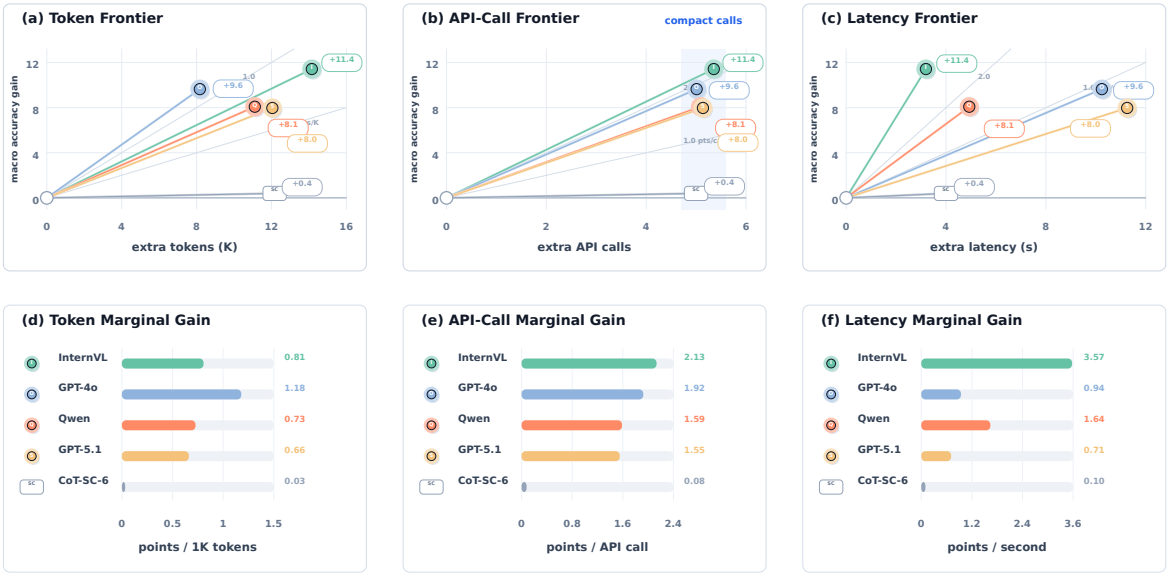

Figure 3: **Cost-effectiveness analysis.** Panels (a)–(c) show macro accuracy gain versus extra tokens, API calls, and latency; panels (d)–(f) show the corresponding marginal gains. EQL-Act requires additional inference budget, but converts it into substantial accuracy gains, whereas CoT-SC-6 spends extra budget with only marginal improvement.

More cost alone does not guarantee better reasoning. CoT-SC-6 provides a useful counterexample: it also consumes additional tokens, API calls, and latency by sampling multiple rationales, but yields only +0.4 percentage points in macro accuracy gain. Its marginal gains are close to zero, with 0.03 points per 1K tokens, 0.08 points per API call, and 0.10 points per second. The key difference is how the budget is organized: CoT-SC-6 spends additional computation on answer-level diversity, whereas EQL-Act spends it on structured evidence-query search, trace filtering, and evidence-level verification.

5 Conclusion

We introduced EQL-Act, an inference-time framework that organizes Scientific VQA as operation-grounded evidence-chain reasoning. Instead of directly generating answers or free-form rationales, EQL-Act routes questions into task-aware EQL operation spaces, expands anchor-initiated EQL traces, and selects answers through trace-level verification. Experiments across three Scientific VQA benchmarks show consistent gains on both proprietary and open-weight MLLMs, while ablations confirm that the improvements come from task-aware operation routing, anchor-guided trace construction, and evidence-aware selection rather than repeated sampling. These findings suggest that explicitly structuring how models seek, maintain, and verify scientific evidence is a promising direction for reliable Scientific VQA.

Limitations

EQL-Act uses a fixed EQL operator interface and a general trace-search procedure, which may not fully capture the fine-grained conventions of highly specialized scientific domains. Extending the operator vocabulary with domain-specific evidence operations could further improve its adaptability. In addition, EQL-Act still relies on the underlying MLLM to instantiate visual observations, so its performance may be affected by weak perception of small details, dense annotations, or fine-grained numerical values. Future work can explore domain-adaptive operators, stronger visual grounding, and more adaptive trace construction.

Acknowledgments

References

Nagaraj N Bhat, Joydeb Mondal, and Srijon Sarkar. 2025. Expertneurons at scivqa-2025: Retrieval augmented vqa with vision language model (ravqa-vlm). In *Proceedings of the Fifth Workshop on Scholarly Document Processing (SDP 2025)*, pages 221–229.

Ekaterina Borisova, Nikolas Rauscher, and Georg Rehm. 2025. SciVQA 2025: Overview of the first scientific visual question answering shared task. In *Proceedings of the Fifth Workshop on Scholarly Document Processing*, pages 182–210. Association for Computational Linguistics.

Ritwick Chaudhry, Sumit Shekhar, Utkarsh Gupta, Pranav Maneriker, Prann Bansal, and Ajay Joshi. 2020. Leaf-qa: Locate, encode & attend for figure question answering. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 3512–3521.

Tanmay Gupta and Aniruddha Kembhavi. 2023. Visual programming: Compositional visual reasoning without training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Kushal Kafle, Brian Price, Scott Cohen, and Christopher Kanan. 2018. Dvqa: Understanding data visualizations via question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5648–5656.

Wonjae Kim, Bokyung Son, and Ildoo Kim. 2021. Vilt: Vision-and-language transformer without convolution or region supervision. In *International Conference on Machine Learning (ICML)*.

Pan Lu, Swaroop Mishra, Tony Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. 2022. Learn to explain: Multimodal reasoning via thought chains for science question answering. In *Advances in Neural Information Processing Systems*.

Pan Lu, Baolin Peng, Hao Cheng, Michel Galley, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, and Jianfeng Gao. 2023. Chameleon: Plug-and-play compositional reasoning with large language models. In *Advances in Neural Information Processing Systems*.

Ahmed Masry, Xuan Long Do, Jia Qing Tan, Shafiq Joty, and Enamul Hoque. 2022. Chartqa: A benchmark for question answering about charts with visual and logical reasoning. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2263–2279.

Ahmed Masry, Megh Thakkar, Aayush Bajaj, Aaryaman Kartha, Enamul Hoque, and Shafiq Joty. 2025. Chartgemma: Visual instruction-tuning for chart reasoning in the wild. In *Proceedings of the 31st International Conference on Computational Linguistics: Industry Track*, pages 625–643.

Nitesh Methani, Pritha Ganguly, Mitesh M Khapra, and Pratyush Kumar. 2020. Plotqa: Reasoning over scientific plots. In *Proceedings of the ieee/cvf winter conference on applications of computer vision*, pages 1527–1536.

Prahitha Movva and Naga Harshita Marupaka. 2025. Enhancing scientific visual question answering through multimodal reasoning and ensemble modeling. In *Proceedings of the Fifth Workshop on Scholarly Document Processing (SDP 2025)*, pages 252–262.

Runqi Qiao, Qiuna Tan, Minghan Yang, Guanting Dong, Peiqing Yang, Shiqiang Lang, Enhui Wan, Xiaowan Wang, Yida Xu, Lan Yang, Chong Sun, Chen Li, Jing Lyu, and Honggang Zhang. 2025. V-thinker: Interactive thinking with images. *arXiv preprint arXiv:2511.04460*.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, and 1 others. 2021. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PmLR.

Hao Shao, Shengju Qian, Han Xiao, Guanglu Song, Zhuofan Zong, Letian Wang, Yu Liu, and Hongsheng Li. 2024. Visual cot: Advancing multi-modal language models with a comprehensive dataset and benchmark for chain-of-thought reasoning. *Advances in Neural Information Processing Systems*, 37:8612–8642.

Jialu Shen, Han Lyu, Suyang Zhong, Hanzheng Li, Haoyi Tao, Nan Wang, Changhong Chen, and Xi Fang. 2026. SpecVQA: A benchmark for spectral understanding and visual question answering in scientific images. *arXiv preprint arXiv:2604.28039*.

Dídac Surís, Amanda Duarte, Alexandre Cour, and Carl Vondrick. 2023. Vipergpt: Visual grounding and reasoning with language models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

Zirui Wang, Mengzhou Xia, Luxi He, Howard Chen, Yitao Liu, Richard Zhu, Kaiqu Liang, Xindi Wu, Haotian Liu, Sadhika Malladi, Alexis Chevalier, Sanjeev Arora, and Danqi Chen. 2024. CharXiv: Charting gaps in realistic chart understanding in multimodal llms. In *Advances in Neural Information Processing Systems*.

Zichen Wen, Boxue Yang, Shuang Chen, Yaojie Zhang, Yuhang Han, Junlong Ke, Cong Wang, Yicheng Fu, Jiawang Zhao, Jiangchao Yao, Xi Fang, Zhen Wang, Henxing Cai, Lin Yao, Zhifeng Gao, Yanhui Hong, Nang Yuan, Yixuan Li, Guojiang Zhao, and 15 others. 2026. Innovator-vl: A multimodal large language model for scientific discovery. *arXiv preprint arXiv:2601.19325*.

Penghao Wu and Saining Xie. 2024. V?: Guided visual search as a core mechanism in multimodal llms. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13084–13094.

Qiang Xu, Shengyuan Bai, Yu Wang, He Cao, Leqing Chen, Yuanyuan Liu, Bin Feng, Zijing Liu, and Yu Li. 2026. ReactBench: A benchmark for topological reasoning in MLLMs on chemical reaction diagrams. *arXiv preprint arXiv:2604.15994*.

Zhengyuan Yang, Linjie Li, Jianfeng Wang, Kevin Lin, Ehsan Azarnasab, Faisal Ahmed, Zicheng Liu, Ce Liu, Michael Zeng, and Lijuan Wang. 2023. MM-REACT: Prompting chatgpt for multimodal reasoning and action. *arXiv preprint arXiv:2303.11381*.

Zhuosheng Zhang, Aston Zhang, Mu Li, Hai Zhao, George Karypis, and Alex Smola. 2023. Multimodal chain-of-thought reasoning in language models. *arXiv preprint arXiv:2302.00923*.

Yilun Zhao, Chengye Wang, Chuhan Li, and Arman Cohan. 2025. Can multimodal foundation models understand schematic diagrams? an empirical study on information-seeking QA over scientific papers. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 18598–18631, Vienna, Austria. Association for Computational Linguistics.

A EQL Operator Inventory

A.1 Global Evidence Operation Set

We define a global evidence operation set, denoted as $\mathcal{O}$, to constrain the actions that an MLLM can perform when constructing an EQL trace. $\mathcal{O}$ is not an answer-type ontology or a task taxonomy. Instead, it specifies a set of standardized evidence-seeking operations for reading scientific figures and collecting support for an answer. Each EQL step must instantiate one valid operation from $\mathcal{O}$, which makes intermediate reasoning more structured and easier to inspect than free-form rationales.

Table 3 lists the current operation inventory. The operations are designed from common failure modes in Scientific VQA: locating the wrong panel or object motivates `LOCATE`; misreading legends or annotations motivates `READ_TEXT`; numerical reading errors motivate `CALIBRATE_MEASURE`; incorrect comparison motivates `COMPARE`; and unsupported answers motivate `RETRIEVE_CONTEXT`, `CHECK_SUPPORT`, and `ABSTAIN_CHECK`.

The operation set is designed by mapping common Scientific VQA failure modes to corrective evidence-seeking actions. For example, errors caused by attending to the wrong panel or visual object are addressed by `LOCATE`; errors caused by misreading legends, labels, or captions are addressed by `READ_TEXT`; errors in axis values or units are addressed by `CALIBRATE_MEASURE` and `VERIFY_UNIT`; and errors from unsupported or under-specified answers are addressed by `RETRIEVE_CONTEXT`, `CHECK_SUPPORT`, and `ABSTAIN_CHECK`. We include `RETURN` as a trace-closure operation rather than an evidence-seeking operation. This design encourages the model to make the underlying scientific operation explicit at each step, instead of producing unconstrained natural-language rationales.

A.2 Machine-readable Dictionary and Validity Constraint

In implementation, the `Operation` column in Table 3 is used as the fixed machine-readable dictionary of valid EQL operations. The corresponding descriptions are used to guide operation-space routing and trace construction. All routed operation spaces and generated EQL traces are sanitized against this dictionary, so that invalid or out-of-vocabulary operations are removed or mapped to the closest valid operation.

During EQL trace construction, every generated step must use an operation from the global set $\mathcal{O}$. Given a routed operation space $\mathcal{O}_q \subseteq \mathcal{O}$, the operation field of each step must satisfy

$$o_t \in \mathcal{O}_q \subseteq \mathcal{O}.$$

If an MLLM proposes an operation outside $\mathcal{O}$, the operation is either mapped to the closest valid operation when the intent is unambiguous, or the step is marked as malformed and penalized during trace scoring. This constraint ensures that EQL traces remain inspectable and comparable across datasets and model backbones.

B Scoring Function Details

This appendix specifies the trace scoring function used in Section 3.5. The score is not intended to be a calibrated probability of correctness. Instead, it is a deterministic utility for pruning clearly unusable traces and constructing a compact candidate portfolio for the final contrastive verifier. We use a linear additive form because the desired properties are monotonic and interpretable: a trace should receive a higher score when it is more complete, better grounded, covers more predicted risks, and better matches the required answer format, while receiving a lower score when it contains malformed operations, unsupported claims, empty outputs, or recorded failures.

Let a completed trace be $\tau = (s_1, \ldots, s_m, \hat{y})$, where each step s_i contains an operation, target, constraints, observation, support status, and possible failure flags. All positive components are normalized to $[0, 1]$, and the penalty term is non-negative. The score is

$$\begin{aligned} S(\tau) = \lambda_c C(\tau) + \lambda_g G(\tau) + \lambda_r R_{\mathcal{F}_q}(\tau) \\ + \lambda_f F_{\Phi_q}(\tau) - \lambda_p P(\tau), \end{aligned} \tag{12}$$

Unless otherwise stated, we use non-negative weights with $\lambda_c + \lambda_g + \lambda_r + \lambda_f = 1$. In our default setting, we set $\lambda_c = 0.25$, $\lambda_g = 0.35$, $\lambda_r = 0.20$, $\lambda_f = 0.20$, and $\lambda_p = 0.50$. We assign the largest positive weight to grounding because unsupported evidence is the most common source of erroneous final answers. The same weights are used across datasets and are not tuned per example.

Completeness. $C(\tau)$ measures whether the trace has filled the fields required by the current query and by the router-predicted execution plan. Let $\mathcal{A}_q$ be the set of required anchors, targets, constraints,

intermediate observations, and answer fields for query q. For each required field $a \in \mathcal{A}_q$, define

$$\mathbf{1}_{\text{filled}}(a, \tau) = \begin{cases} 1, & \text{if } a \text{ is explicitly instantiated} \\ & \text{in } \tau, \\ 0, & \text{otherwise.} \end{cases} \tag{13}$$

Then

$$C(\tau) = \frac{1}{|\mathcal{A}_q|} \sum_{a \in \mathcal{A}_q} \mathbf{1}_{\text{filled}}(a, \tau). \tag{14}$$

A trace with missing targets, missing constraints, or no final answer therefore receives a lower completeness score.

Grounding. $G(\tau)$ measures whether the trace's observations support the claims made in the trace and the final answer. Let $\mathcal{U}(\tau)$ be the set of atomic claims extracted from the trace, including intermediate observations and final answer claims. Each claim $u \in \mathcal{U}(\tau)$ is assigned a support label $\ell(u) \in \{\text{SUPPORTED}, \text{PARTIALLY SUPPORTED}, \text{UNSUPPORTED}, \text{UNKNOWN}\}$. We map these labels to numeric values:

$$g(u) = \begin{cases} 1.0, & \ell(u) = \text{SUPPORTED}, \\ 0.5, & \ell(u) = \text{PARTIALLY SUPPORTED}, \\ 0.0, & \ell(u) = \text{UNSUPPORTED}, \\ 0.0, & \ell(u) = \text{UNKNOWN}. \end{cases} \tag{15}$$

The grounding score is

$$G(\tau) = \frac{1}{|\mathcal{U}(\tau)|} \sum_{u \in \mathcal{U}(\tau)} g(u). \tag{16}$$

If $\mathcal{U}(\tau)$ is empty, we set $G(\tau) = 0$.

Risk Coverage. $R_{\mathcal{F}_q}(\tau)$ measures whether the trace addresses the failure or ambiguity risks predicted by the router. Let $\mathcal{F}_q$ be the set of router-predicted risks for query q, such as ambiguity, missing evidence, multi-hop dependency, temporal mismatch, entity mismatch, or answer-format mismatch. For each risk $f \in \mathcal{F}_q$, define

$$\rho(f, \tau) = \begin{cases} 1, & \text{if } \tau \text{ explicitly checks} \\ & \text{or resolves } f, \\ 0.5, & \text{if } \tau \text{ partially addresses } f, \\ 0, & \text{otherwise.} \end{cases} \tag{17}$$

Then

$$R_{\mathcal{F}_q}(\tau) = \frac{1}{|\mathcal{F}_q|} \sum_{f \in \mathcal{F}_q} \rho(f, \tau). \tag{18}$$

If the router predicts no active risks, we set $R_{\mathcal{F}_q}(\tau) = 1$, since there is no risk requirement left uncovered.

Format Compatibility. $F_{\Phi_q}(\tau)$ measures whether the candidate answer is compatible with the answer format required by the query. Let Φ_q denote the expected answer schema, including answer type, cardinality, unit, normalization constraints, and abstention requirement if applicable. We decompose this into format checks $\mathcal{M}_{\Phi_q}$ and compute

$$F_{\Phi_q}(\tau) = \frac{1}{|\mathcal{M}_{\Phi_q}|} \sum_{\phi \in \mathcal{M}_{\Phi_q}} \mathbf{1}_{\text{match}}(\phi, \tau). \tag{19}$$

For example, $\mathbf{1}_{\text{match}}$ checks whether a yes/no question receives a binary answer, whether a numerical answer contains the required unit, whether a list answer has the expected cardinality, and whether an unanswerable case is represented by abstention rather than by an unsupported guess.

Penalty. $P(\tau)$ aggregates hard and soft failure signals. We use

$$\begin{aligned} P(\tau) = {} & w_{\text{json}} p_{\text{json}} + w_{\text{op}} p_{\text{op}} + w_{\text{sup}} p_{\text{sup}} \\ & + w_{\text{empty}} p_{\text{empty}} + w_{\text{fail}} p_{\text{fail}}. \end{aligned} \tag{20}$$

where each $p_{\cdot} \in \{0, 1\}$ indicates whether the trace contains the corresponding failure type. Specifically, p_{json} marks invalid JSON or unparsable trace structure, p_{op} marks illegal or invalid operations, p_{sup} marks unsupported final answers, p_{empty} marks empty observations or empty final outputs, and p_{fail} marks recorded execution failures such as transient API errors or tool failures. In the default setting, we use $w_{\text{json}} = 1.0$, $w_{\text{op}} = 1.0$, $w_{\text{sup}} = 1.0$, $w_{\text{empty}} = 0.5$, and $w_{\text{fail}} = 0.5$. Traces with invalid JSON, illegal operations, invalid support status, or transient API errors are removed before Top-K selection, regardless of their scalar score.

Filtering and Top-K selection. After computing $S(\tau)$, we discard traces that violate hard validity constraints or fall below a minimum score threshold θ. For the remaining traces, we rank by $(S(\tau), \text{conf}(\tau))$ in descending lexicographic order and keep the Top-K traces. The scalar score is therefore used only to form a reliable and diverse verification portfolio; the final answer is still selected by the contrastive verifier over trace-level evidence rather than by direct score maximization or answer-string voting.

C Benchmark Evaluation Details

C.1 Benchmark Details

Table 4 summarizes the benchmark settings used in our evaluation. The three benchmarks cover complementary Scientific VQA scenarios: schematic-diagram information seeking, spectral understanding, and topological reasoning over reaction diagrams. To align with the input formulation of EQL-Act, we describe each benchmark input as an image-question pair, optionally augmented with paper context.

C.2 Main Result Details

The detailed results in Tables 5–7 show that EQL-Act consistently improves multimodal models across datasets, task formats, and evaluation dimensions. The gains are not restricted to a specific benchmark or backbone model. For example, on MISS-QA, EQL-Act improves GPT-5.1 from 70.60 to 81.10 and GPT-4o from 63.00 to 77.64. On ReactBench, it improves Gemini-2.5-Flash from 64.29 to 74.75, while on SpecVQA it improves Qwen3-VL-8B from 51.58 to 59.56. These results suggest that EQL-Act strengthens the ability of multimodal models to organize visual evidence, perform structured reasoning, and make more reliable decisions in scientific VQA tasks.

On MISS-QA, EQL-Act yields the most pronounced improvements, especially on question types that require understanding method design, implementation details, literature context, and experimental evidence. EQL-Act w/ GPT-5.1 achieves the best TestMini accuracy of 81.10 and ranks first on the Design, Implementation, Literature, and Experiment dimensions. In particular, its Experiment accuracy increases from 70.21 to 87.05, corresponding to a gain of 16.84 points. The improvements are also substantial for smaller backbones: EQL-Act raises Phi-3.5-Vision from 30.30 to 46.30 on TestMini. This indicates that EQL-Act benefits both strong proprietary models and weaker open-weight models by compensating for their limitations in scientific reasoning. We also observe that Gemini-2.5-Flash already performs strongly on Other and Unanswerable questions, which leaves less room for improvement and leads to a slight drop on Other after applying EQL-Act. Nevertheless, its overall TestMini accuracy still improves from 67.30 to 73.80.

On ReactBench, EQL-Act mainly improves abilities related to visual localization, process tracing, and complex reasoning. EQL-Act w/ Gemini-2.5-Flash achieves the best average accuracy of 74.75, followed by EQL-Act w/ GPT-5.1 with 74.12. Notably, EQL-Act w/ InternVL3.5 improves from 52.72 to 73.77, yielding a large gain of 21.05 points. Dimension-level results further show that the gains are particularly strong on Localization and Reasoning. For instance, InternVL3.5 improves from 22.22 to 63.96 on Reasoning, and Qwen3-VL-8B improves from 31.67 to 57.32. These results indicate that EQL-Act is especially effective when the model must integrate evidence across visual regions, trace intermediate steps, and make task-level judgments. In contrast, some models show small drops on Extraction, suggesting that additional reasoning may not always benefit tasks dominated by direct information retrieval.

On SpecVQA, EQL-Act provides stable improvements across languages and reasoning levels. Although Gemini-2.5-Flash is already a strong baseline, EQL-Act further improves its Overall accuracy from 72.74 to 74.65, achieving the best result among all methods. EQL-Act also improves GPT-5.1 from 65.78 to 71.81. More importantly, the gains are clear on L1 reasoning questions in both English and Chinese. For example, Qwen3-VL-8B improves from 45.94 to 56.47 on En-L1 and from 48.26 to 61.18 on Zh-L1, while InternVL3.5 improves from 49.02 to 61.76 on Zh-L1. These findings suggest that EQL-Act does not merely enhance surface-level visual recognition or language matching; instead, it improves evidence organization and reasoning across languages and difficulty levels. Overall, the three tables demonstrate that EQL-Act has strong model-level and task-level generalization, particularly for scientific multimodal QA tasks that require complex reasoning, experimental understanding, and visual evidence localization.

D Prompt Templates

This appendix reports the prompt templates used by EQL-Act. The inference pipeline uses four prompts: an LLM router, a semantic anchor generator, a stepwise EQL particle expander, and a Top-K contrastive verifier. We also report the LLM-as-judge prompt used for evaluation. The judge prompt is not part of the inference pipeline.

All prompt templates are written with placeholders such as `{dataset}`, `{question}`, `{router_output_json}`, and

`{selected_particles_json}`. At inference time, these placeholders are replaced with the corresponding benchmark instance, dataset context, operation vocabulary, controller state, or selected traces. To make the procedure reproducible, every prompt requires valid JSON output and forbids markdown wrapping. This design lets the controller parse each response deterministically, validate operation names against the admissible EQL vocabulary, and pass structured traces to later stages.

The prompts are intentionally separated by role. The router and anchor generator do not answer the question; they only restrict the operation space and define evidence entry points. The particle prompt executes one operation at a time and exposes local observations, missing evidence, support status, and a candidate answer only when the current step is sufficient. The contrastive verifier then compares the selected traces by grounded evidence rather than by answer frequency. This separation is important because EQL-Act is designed to control the evidence path of scientific VQA, not simply to elicit a longer free-form rationale.

D.1 Inference Prompts

Figures 4–7 present the four inference prompts used by EQL-Act. Figure 4 shows the router prompt, which selects the admissible operation space. Figure 5 shows the semantic anchor prompt, which identifies question-specific evidence entry points. Figure 6 shows the stepwise EQL particle expansion prompt, which executes one operation at a time and updates the trace state. Figure 7 shows the Top-K contrastive verifier prompt, which compares candidate traces by evidence support.

D.2 Evaluation Prompt

Figure 8 reports the LLM-as-judge evaluation prompt. This prompt is used only when exact string matching is insufficient. It receives the question, optional context, gold answer, model prediction, and answer format, and returns a structured correctness judgment. It is never used to generate EQL-Act predictions.

Operation	Role	Code-level definition	Typical routed use
LOCATE	Visual grounding	Locate a target region, panel, node, peak, curve, mask, or module.	Used across all datasets to ground the relevant visual object, panel, highlighted region, spectrum component, or reaction node before further evidence operations.
READ_TEXT	Text reading	Read visible text, labels, legends, annotations, or captions.	Triggered for labels, legends, axis text, annotations, masked-region labels, and visible textual cues in scientific figures.
INSPECT_RELATION	Relation inspection	Inspect arrows, links, relations, dependencies, or flow direction.	Used for reaction paths, schematic dependencies, input-output relations, neighboring entities, and relation direction errors.
TRACE_PROCESS	Process tracing	Trace a reaction path, process path, or multi-hop relation.	Used when questions require following pathways, generation routes, process chains, or multi-hop scientific relations.
CALIBRATE_MEASURE	Measurement	Read and calibrate a measured value from axis, tick, peak, or label.	Used for numerical values, counts, spectra, axes, peaks, experimental conditions, and value-sensitive questions.
COMPARE	Comparison	Compare peaks, curves, candidates, trends, or values.	Used for higher/lower, larger/smaller, strongest/weakest, ranking, similarity, trend, curve, peak, and candidate comparison.
VERIFY_UNIT	Format and unit check	Verify the unit, scope, and answer format.	Used for unit-sensitive measurements, axis-based reading, numeric answers, answer-format constraints, and scope verification.
RETRIEVE_CONTEXT	Context retrieval	Retrieve supporting evidence from paper context.	Primarily used in MISS-QA to retrieve relevant paper context, captions, or surrounding textual evidence beyond the diagram itself.
CHECK_SUPPORT	Support checking	Check whether the evidence supports the answer.	Used to verify whether the candidate answer is grounded in image evidence, text context, or both.
ABSTAIN_CHECK	Answerability check	Check whether evidence is insufficient and the model should abstain.	Used for unsupported or under-specified cases, especially MISS-QA questions where the correct behavior may be to answer `I do not know.`
RETURN	Terminal action	Return the final answer.	A terminal operation appended by the router to close an EQL trace once sufficient evidence has been collected or abstention is justified.

Table 3: **Global EQL operation inventory.** Each EQL step must instantiate one valid operation and record its target, constraints, observation, missing evidence, local confidence, support status, and stopping decision.

Dataset	Domain	#Q	Split	Input	Metric
MISS-QA	Schematics	1,500†	Official	$I + q + C$	LLM judge
SpecVQA	Spectra / plots	6,200	Official	$I + q$	Judge + tolerance
ReactBench	Reaction diagrams	1,618	Official	$I + q$	Normalized EM

Table 4: Benchmark statistics and evaluation protocols. Here, I denotes the scientific image, q denotes the question, and C denotes optional paper context. MISS-QA evaluates information-seeking question answering over schematic diagrams with paper context; we use the official MISS-QA evaluation set containing 1,500 expert-annotated examples over 465 scientific papers. SpecVQA evaluates spectral understanding over 620 scientific figures and 6,200 QA pairs. ReactBench evaluates topological reasoning over chemical reaction diagrams with 1,618 expert-annotated QA pairs.

Model	Params	Design	Impl.	Lit.	Exp.	Other	Unans.	TestMini. ↑
Proprietary Multimodal Large Language Models								
GPT-5.1	–	77.60	69.84	68.83	70.21	45.36	49.39	70.60
GPT-4o	–	73.90	67.50	67.20	65.80	38.50	34.70	63.00
Gemini-2.5-Flash	–	72.20	63.40	57.20	57.80	**87.20**	**88.00**	67.30
Claude-4-Sonnet	–	79.32	70.35	72.84	73.16	54.97	49.81	71.35
Claude-3.5-Sonnet	–	78.52	69.44	**73.73**	**75.19**	58.64	53.20	71.87
Open-weight Multimodal Large Language Models								
Qwen2.5-VL	72B	72.23	61.34	62.55	62.73	43.21	33.98	59.71
Qwen3-VL	30B	63.41	51.23	52.38	60.17	43.55	53.30	57.09
Qwen3-VL	8B	60.24	45.88	43.62	48.97	54.83	50.14	55.40
InternVL2.5	38B	56.50	49.50	50.20	51.60	50.30	52.00	51.60
InternVL3.5	8B	50.41	36.14	42.05	45.13	40.76	38.08	45.80
Phi-3.5-Vision	4.2B	18.80	22.60	29.60	23.30	59.90	81.30	30.30
DeepSeek-VL2	16B	20.21	19.88	26.74	28.62	42.19	51.93	29.86
LLaVA-Next-Vicuna	13B	18.22	16.63	27.95	25.20	30.17	68.84	23.62
Scientific VQA-related Methods								
CoT w/ Qwen3-VL	8B	64.63	53.12	49.39	58.66	39.53	49.81	56.01
V-Thinker w/ Qwen2.5-VL	7B	59.52	56.18	58.62	60.42	41.91	34.59	56.83
ChartGemma	7B	68.01	59.54	61.92	43.77	45.39	48.47	58.94
Innovator-VL	8B	66.14	58.40	60.25	62.22	43.81	35.94	58.21
E-SVQA w/ Qwen3-VL	8B	66.92	57.26	58.23	66.38	43.89	37.50	58.40
ExpertNeurons w/ InternVL3	14B	58.42	55.20	47.38	48.19	40.39	40.18	55.49
Our EQL-Act								
EQL-Act w/ GPT-5.1	–	**87.63** ↑**10.03**	**82.71** ↑**12.87**	**73.96** ↑**5.13**	**87.05** ↑**16.84**	76.28 ↑**30.92**	77.78 ↑**28.39**	**81.10** ↑**10.50**
EQL-Act w/ GPT-4o	–	**80.93** ↑**7.03**	**79.94** ↑**12.44**	70.06 ↑**2.86**	74.11 ↑**8.31**	72.70 ↑**34.20**	68.82 ↑**34.12**	**77.64** ↑**14.64**
EQL-Act w/ Gemini-2.5-Flash	–	**80.35** ↑**8.15**	70.83 ↑**7.43**	64.99 ↑**7.79**	64.14 ↑**6.34**	**85.58**↓1.62	**91.91** ↑**3.91**	73.80 ↑**6.50**
EQL-Act w/ Qwen3-VL	8B	68.92 ↑**8.68**	59.48 ↑**13.60**	67.37 ↑**23.75**	52.04 ↑**3.07**	48.66↓6.17	63.32 ↑**13.18**	63.70 ↑**8.30**
EQL-Act w/ InternVL3.5	8B	44.71↓5.70	53.53 ↑**17.39**	42.84 ↑**0.79**	46.93 ↑**1.80**	63.27 ↑**22.51**	61.26 ↑**23.18**	51.90 ↑**6.10**
EQL-Act w/ Phi-3.5-Vision	4.2B	35.74 ↑**16.94**	40.44 ↑**17.84**	42.32 ↑**12.72**	38.17 ↑**14.87**	63.16 ↑**3.26**	83.41 ↑**2.11**	46.30 ↑**16.00**

Table 5: **Detailed MISS-QA results by question type.** All results are reported in accuracy (%). TestMini denotes the overall accuracy used in Table 1. **Red** and **Blue** mark the best and second-best scores in each column, respectively.

Model	Params	Localization	Extraction	Tracing	Reasoning	Avg. ↑
Proprietary Multimodal Large Language Models						
GPT-5.1	–	45.38	79.23	84.00	58.33	66.74
GPT-4o	–	44.07	79.71	82.04	49.50	63.83
Gemini-2.5-Flash	–	51.44	78.96	80.03	46.73	64.29
Claude-4-Sonnet	–	51.75	80.63	89.72	54.42	69.13
Claude-3.5-Sonnet	–	50.78	81.88	**92.22**	49.50	68.60
Open-weight Multimodal Large Language Models						
Qwen2.5-VL	72B	43.66	87.11	79.06	50.89	65.18
Qwen3-VL	30B	44.12	74.03	75.43	44.94	59.63
Qwen3-VL	8B	42.69	84.62	76.00	31.67	58.74
InternVL2.5	38B	36.75	69.92	64.59	21.58	48.21
InternVL3.5	8B	42.95	84.38	61.33	22.22	52.72
Phi-3.5-Vision	4.2B	21.80	69.75	29.34	38.12	39.75
DeepSeek-VL2	16B	28.62	78.50	8.38	28.71	36.05
LLaVA-Next-Vicuna	13B	13.77	39.13	1.80	28.22	20.73
Scientific VQA-related Methods						
CoT w/ Qwen3-VL	8B	45.27	70.82	77.34	49.57	60.75
V-Thinker w/ Qwen2.5-VL	7B	46.11	72.73	75.86	50.26	61.24
ChartGemma	7B	49.97	81.03	77.29	52.07	65.09
Innovator-VL	8B	45.92	73.84	74.95	56.77	62.87
E-SVQA w/ Qwen3-VL	8B	42.83	69.36	74.88	51.37	59.61
ExpertNeurons w/ InternVL3	14B	38.09	83.74	84.22	46.83	63.22
Our EQL-Act						
EQL-Act w/ GPT-5.1	–	**64.23** ↑**18.85**	**89.23** ↑**10.00**	78.03↓5.97	**64.97** ↑**6.64**	**74.12** ↑**7.38**
EQL-Act w/ GPT-4o	–	52.41 ↑**8.34**	79.13↓0.58	87.06 ↑**5.02**	**65.92** ↑**16.42**	71.13 ↑**7.30**
EQL-Act w/ Gemini-2.5-Flash	–	**66.92** ↑**15.48**	83.08 ↑**4.12**	**93.73** ↑**13.70**	55.27 ↑**8.54**	**74.75** ↑**10.46**
EQL-Act w/ Qwen3-VL	8B	46.78 ↑**4.09**	80.77↓3.85	81.89 ↑**5.89**	57.32 ↑**25.65**	66.69 ↑**7.95**
EQL-Act w/ InternVL3.5	8B	60.77 ↑**17.82**	**87.31** ↑**2.93**	83.04 ↑**21.71**	63.96 ↑**41.74**	73.77 ↑**21.05**
EQL-Act w/ Phi-3.5-Vision	4.2B	26.15 ↑**4.35**	67.69↓2.06	36.00 ↑**6.66**	43.33 ↑**5.21**	43.29 ↑**3.54**

Table 6: **Detailed ReactBench results by task dimension.** All results are reported in accuracy (%). Avg. denotes the ReactBench average used in Table 1. **Red** and **Blue** mark the best and second-best scores in each column, respectively.

Model	Params	En-L0	En-L1	En Avg.	Zh-L0	Zh-L1	Zh Avg.	Overall ↑
Proprietary Multimodal Large Language Models								
GPT-5.1	–	67.76	63.47	65.62	68.99	62.90	65.95	65.78
GPT-4o	–	65.75	57.85	61.80	63.64	55.88	59.76	60.78
Gemini-2.5-Flash	–	**73.95**	**70.43**	**72.19**	**74.73**	**71.85**	**73.29**	**72.74**
Claude-4-Sonnet	–	59.47	52.82	56.15	59.57	55.65	57.61	56.88
Claude-3.5-Sonnet	–	60.34	52.26	56.30	57.58	50.58	54.08	55.19
Open-weight Multimodal Large Language Models								
Qwen2.5-VL	72B	62.42	53.98	58.20	64.04	55.36	59.70	58.95
Qwen3-VL	30B	61.01	52.83	56.92	62.02	53.58	57.80	57.36
Qwen3-VL	8B	55.76	45.94	50.85	56.36	48.26	52.31	51.58
InternVL2.5	38B	60.90	52.46	56.68	62.12	52.40	57.26	56.97
InternVL3.5	8B	57.78	49.80	53.79	60.40	49.02	54.71	54.25
Phi-3.5-Vision	4.2B	39.15	31.25	35.20	35.98	30.06	33.02	34.11
DeepSeek-VL2	16B	46.57	31.83	39.20	40.92	30.79	35.86	37.53
LLaVA-Next-Vicuna	13B	34.55	27.17	30.86	31.42	26.10	28.76	29.81
Scientific VQA-related Methods								
CoT w/ Qwen3-VL	8B	58.18	47.82	53.00	58.48	49.00	53.74	53.37
V-Thinker w/ Qwen2.5-VL	7B	56.40	48.32	52.36	57.21	49.03	53.12	52.74
ChartGemma	7B	63.64	58.04	60.84	64.75	53.65	59.20	60.02
Innovator-VL	8B	58.43	49.69	54.06	59.09	50.47	54.78	54.42
E-SVQA w/ Qwen3-VL	8B	60.29	51.99	56.14	61.56	51.48	56.52	56.33
ExpertNeurons w/ InternVL3	14B	66.21	59.49	62.85	66.36	57.22	61.79	62.32
Our EQL-Act								
EQL-Act w/ GPT-5.1	–	**73.33** ↑**5.57**	69.41 ↑**5.94**	71.37 ↑**5.75**	72.73 ↑**3.74**	71.76 ↑**8.86**	72.25 ↑**6.30**	71.81 ↑**6.03**
EQL-Act w/ GPT-4o	–	71.52 ↑**5.77**	63.53 ↑**5.68**	67.52 ↑**5.72**	73.94 ↑**10.30**	62.35 ↑**6.47**	68.15 ↑**8.39**	67.83 ↑**7.05**
EQL-Act w/ Gemini-2.5-Flash	–	72.73↓1.22	**72.94** ↑**2.51**	**72.83** ↑**0.64**	**80.00** ↑**5.27**	**72.94** ↑**1.09**	**76.47** ↑**3.18**	**74.65** ↑**1.91**
EQL-Act w/ Qwen3-VL	8B	61.21 ↑**5.45**	56.47 ↑**10.53**	58.84 ↑**7.99**	59.39 ↑**3.03**	61.18 ↑**12.92**	60.29 ↑**7.98**	59.56 ↑**7.98**
EQL-Act w/ InternVL3.5	8B	60.91 ↑**3.13**	58.24 ↑**8.44**	59.57 ↑**5.78**	64.55 ↑**4.15**	61.76 ↑**12.74**	63.16 ↑**8.45**	61.36 ↑**7.11**
EQL-Act w/ Phi-3.5-Vision	4.2B	41.82 ↑**2.67**	47.06 ↑**15.81**	44.44 ↑**9.24**	33.94↓2.04	34.12 ↑**4.06**	34.03 ↑**1.01**	39.23 ↑**5.12**

Table 7: **Detailed SpecVQA results by language and reasoning level.** All results are reported in accuracy (%). Overall denotes the SpecVQA score used in Table 1. **Red** and **Blue** mark the best and second-best scores in each column, respectively.

Router Prompt

```
[System]
You are an EQL operation-space router for scientific visual question answering.

Your task is not to answer the question.
Your task is to decide which EQL operations are needed to solve the question.

Return valid JSON only.
Do not wrap the JSON in markdown.

[User]
Dataset:
{dataset}

Question:
{question}

Global EQL operations:
{ops_json}

Rule-router proposal:
{rule_router_output_json}

Return one JSON object with this schema:
{
  "candidate_ops": [
    "operation names from Global EQL operations"
  ],
  "risk_flags": [
    "axis|attribute|calculation|comparison|context|counting|ranking|reasoning|semantic
    ↪ |subplot|tracing"
  ],
  "answer_format": "number|yes_no|unit_value|grounded_text|short_phrase",
  "rationale": "one short sentence explaining the routing decision"
}

Rules:
- Do not answer the question.
- Use only operation names from Global EQL operations.
- Keep candidate_ops compact and task-specific.
- Preserve rule-proposed operations unless they are clearly irrelevant.
- Add an operation only when it corresponds to a concrete evidence need.
- RETURN may be omitted; the controller will append it if needed.
- risk_flags must describe failure risks that later traces should cover.
- answer_format must describe the expected final answer form.
- Prefer evidence operations over broad task labels.
- Return JSON only.

Routing guidance:
- If the question specifies a panel, subplot, inset, or region, include a subplot/
    ↪ scope risk and operations for localization or reading labels.
- If the question asks for axes, ticks, units, values, shifts, or measurements,
    ↪ include measurement/unit operations and an axis or calculation risk.
- If the question asks for highest, lowest, strongest, weakest, second, ranking, or
    ↪ comparison, include COMPARE and a ranking or comparison risk.
- If the question asks for a reaction path, previous/next step, precursor, successor,
    ↪ or process direction, include relation/process operations and a tracing risk.
- If the question requires paper context, answerability, or explicit support, include
    ↪ context/support/abstention operations and a context risk.
```

Figure 4: Router prompt used to select a compact set of admissible EQL operations before trace expansion.

Semantic Anchor Prompt

```
[System]
You are an anchor generator for EQL-Act, a scientific visual question answering method.

Your task is not to answer the que
stion.
Your task is to generate semantic anchors for multiple evidence-query traces.

Return valid JSON only.
Do not wrap the JSON in markdown.

[User]
Dataset:
{dataset}

Question:
{question}

Router output:
{router_output_json}

Available EQL operations:
{available_ops_json}

A semantic anchor is the starting evidence focus of one particle.
It tells a particle where to begin its trace, such as a panel, axis, peak,
curve, mask, entity, relation, context span, or answerability condition.

Return one JSON object with this schema:
{
  "anchors": [
    {
      "anchor_id": "short_snake_case_id",
      "cue": "exact phrase or semantic cue from the question",
      "intent": "what evidence this anchor should inspect first",
      "compatible_ops": [
        "operations from router_output.candidate_ops only"
      ],
      "priority": 0.0,
      "source": "llm"
    }
  ]
}

Rules:
- Do not answer the question.
- Generate 2 to 5 anchors.
- Each anchor must cover a distinct evidence entry point.
- Every compatible_ops item must appear in router_output.candidate_ops.
- Do not include RETURN in compatible_ops, because anchors define starting operations.
- Do not invent operations outside Available EQL operations.
- Prefer concrete anchors over abstract reasoning labels.
- The highest-priority anchor should handle the strongest scope constraint, such as panel, mask, axis,
    ↪ entity, relation, or context.
- If the question asks for comparison, ranking, highest, lowest, strongest, weakest, or difference, include
    ↪ a comparison anchor.
- If the question requires axes, units, numerical values, shifts, or measurements, include a measurement or
    ↪ unit anchor.
- If the question requires relation direction, process order, previous/next item, or pathway tracing,
    ↪ include a relation or process anchor.
- If the question requires context support or answerability, include a support or abstention anchor.
- priority must be between 0 and 1.
- source must be "llm".
- Return JSON only.

Example of the intended behavior:
Question: "In panel (b), which peak has the highest intensity?"
Possible anchors:
- panel_scope: cue "panel (b)", compatible_ops ["LOCATE", "READ_TEXT"]
- peak_candidates: cue "peak", compatible_ops ["LOCATE", "READ_TEXT", "COMPARE"]
- intensity_ranking: cue "highest intensity", compatible_ops ["COMPARE"]
```

Figure 5: Semantic anchor prompt used to identify evidence entry points and question-specific visual or contextual anchors.

Stepwise EQL Particle Expansion Prompt

```
[System]
You are expanding one EQL particle for scientific visual question answering.

At this step, you must execute exactly one EQL operation.
Do not write free-form chain-of-thought.
Return valid JSON only.
Do not wrap the JSON in markdown.

[User]
Dataset:
{dataset}

Question:
{question}

Dataset context:
{dataset_context}

Router output:
{router_output_json}

Anchor:
{anchor_json}

Particle:
{
  "particle_id": "{particle_id}",
  "start_op": "{start_op}",
  "current_depth": {current_depth},
  "max_steps": {max_steps}
}

Existing EQL trace:
{existing_trace_json}

Current particle state:
{state_json}

Current admissible operations:
{admissible_ops_json}

Return one JSON object with this schema:
{
  "operation": "one operation from Current admissible operations",
  "target": "what is being inspected at this step",
  "constraints": "scope, panel, object class, unit, relation, or context constraint",
  "observation": "short grounded observation from image/question/context",
  "missing": "what remains uncertain after this step",
  "local_confidence": 0.0,
  "candidate_answer": "empty unless this step supports a concise answer",
  "support_status": "supported|partial|unsupported",
  "failure_modes": [],
  "stop": false
}

Rules:
- Choose exactly one operation from Current admissible operations.
- If current_depth is 0, operation must equal start_op.
- Execute only the selected operation; do not combine multiple operations in one step.
- target must name the concrete object, panel, axis, peak, entity, relation, mask, or context span being inspected.
- constraints must state the relevant scope, such as panel, axis, unit, object class, relation direction, or context boundary.
- observation must be grounded in the image, question, or provided context.
- missing must state what evidence remains unresolved after this step.
- candidate_answer must stay empty unless the current evidence is sufficient to support a final answer.
- support_status must be one of supported, partial, or unsupported.
- local_confidence must be between 0 and 1.
- Set stop=true only if the trace is complete, the operation is RETURN, or the evidence is insufficient and the system should not answer.
- If evidence is insufficient, use support_status="unsupported"; for answerability tasks, candidate_answer may be "I do not know."
- Follow router_output.answer_format for any candidate_answer.
- Return JSON only.

Operation discipline:
- LOCATE should only identify the relevant visual region or object.
- READ_TEXT should only read visible labels, legends, captions, or annotations.
- INSPECT_RELATION should only inspect links, arrows, dependencies, or relation direction.
- TRACE_PROCESS should only follow a process, reaction path, or multi-hop relation.
- CALIBRATE_MEASURE should only read or calibrate values from axes, ticks, peaks, or labels.
- COMPARE should only compare candidates, trends, curves, peaks, or values.
- RETRIEVE_CONTEXT should only retrieve supporting context.
- CHECK_SUPPORT should only judge whether evidence supports the candidate answer.
- ABSTAIN_CHECK should only judge whether evidence is insufficient.
- VERIFY_UNIT should only verify unit, scope, and answer format.
- RETURN should only return the final concise answer.
```

Figure 6: Stepwise EQL particle expansion prompt used to execute one evidence operation at a time and update the structured trace.

Top-K Contrastive Verifier Prompt

```
[System]
You are the EQL-Act contrastive verifier.

Your task is to compare selected evidence traces and choose the answer best supported by evidence.
Do not invent a new trace.
Do not use answer frequency as the main criterion.

Return valid JSON only.
Do not wrap the JSON in markdown.

[User]
Dataset:
{dataset}

Question:
{question}

Dataset context:
{dataset_context}

Router output:
{router_output_json}

Selected particles:
{selected_particles_json}

Evidence risks to verify:
{risk_flags_json}

Answer format:
{answer_format}

Return one JSON object with this schema:
{
  "selected_particle": "particle_id from Selected particles",
  "final_answer": "concise final answer",
  "confidence": 0.0,
  "answerability": "supported|partial|unanswerable",
  "decisive_differences": [
    "short evidence-based reason why the selected trace is better"
  ],
  "rejected": [
    {
      "particle_id": "particle_id from Selected particles",
      "reason": "why this trace was not selected"
    }
  ]
}

Rules:
- Do not generate a new trace.
- Select only one particle from Selected particles.
- Compare traces by anchor, start_op, operation sequence, target, constraints, observation, missing, support_status, failure_modes, and answer format.
- Prefer the trace that correctly grounds the required panel, axis, unit, object, relation, context, or answerability condition.
- Penalize wrong panel, wrong axis, wrong unit, wrong relation direction, wrong scope, unsupported context, empty observations, unresolved critical missing evidence, and over-answering.
- Do not choose an answer merely because it appears in more traces.
- If all selected traces are unsupported, answerability must be "unanswerable".
- For context-support or answerability tasks, if no selected trace is supported by explicit image/context evidence, final_answer must be "I do not know."
- final_answer must obey Answer format.
- confidence must be between 0 and 1.
- decisive_differences must refer to evidence differences among selected particles, not generic preferences.
- rejected should include every non-selected particle when possible.
- Return JSON only.

Dataset-specific checks:
- For scientific plots, prefer traces that correctly establish panel, axis, unit, target item, and comparison scope.
- For reaction or process diagrams, prefer traces that correctly follow relation direction and distinguish entities, conditions, products, and steps.
- For context-grounded questions, prefer traces that explicitly retrieve and check supporting context, and abstain when support is insufficient.
```

Figure 7: Top-K contrastive verifier prompt used to compare selected traces by grounded evidence rather than answer frequency.

LLM-as-judge Evaluation Prompt

```
[System]
You are an impartial evaluator for scientific visual question answering.

Your task is to judge whether the model prediction matches the ground-truth answer.
You must not solve the original question from scratch.
You must compare the prediction against the ground truth under the dataset-specific scoring rules.

Return valid JSON only.
Do not wrap the JSON in markdown.

[User]
Dataset:
{dataset}

Question:
{question}

Optional context or caption:
{context}

Ground-truth answer:
{gold_answer}

Model prediction:
{prediction}

Answer format:
{answer_format}

Return one JSON object with this schema:
{
  "correct": true,
  "score": 1.0,
  "error_type": "none|wrong_value|wrong_unit|wrong_entity|wrong_relation|wrong_scope|unsupported|over_answer|format_error|empty_answer|other",
  "explanation": "brief reason for the judgment"
}

Scoring rules:
- Score 1.0 if the prediction is fully correct.
- Score 0.5 if the prediction captures the main answer but is incomplete, slightly underspecified, or contains a minor non-critical wording issue.
- Score 0.0 if the prediction is wrong, unsupported, empty, contradicts the ground truth, or answers a different question.
- Set correct=true only when score is 1.0.
- Keep explanation short and evidence-focused.

General equivalence rules:
- Ignore harmless wording differences.
- Ignore capitalization and punctuation differences unless they change a chemical label, option label, unit, or entity name.
- Treat equivalent numbers written in words and digits as the same.
- Do not give credit for long explanations if the final answer is wrong.
- Do not penalize extra explanation if the final answer is clearly correct and not contradictory.
- If the prediction contains multiple conflicting answers, judge the final explicit answer; if no final answer is identifiable, score 0.0.

Numeric and unit rules:
- If the ground truth is numeric, accept a prediction within 5 percent relative error unless the dataset requires exact counting.
- For exact counting questions, require the same count.
- If the ground truth includes a unit, the prediction must use the same unit or a clearly equivalent converted unit.
- If the numeric value is correct but the unit is missing or wrong, assign score 0.5 for minor unit omission and 0.0 for a misleading or incompatible unit.

Dataset-specific rules:
- MISS-QA:
  - Score 1.0 if the prediction matches the ground truth and is supported by the given context or the expected abstention behavior.
  - Score 0.5 if it captures part of the ground truth but is incomplete or slightly imprecise.
  - Score 0.0 if it uses unsupported outside knowledge, over-answers an unanswerable example, or fails to abstain when the correct answer is "I do not know."
- SpecVQA:
  - For numeric chart or plot answers, use the numeric and unit rules above.
  - For non-numeric answers, judge semantic equivalence and ignore minor wording differences.
  - Penalize wrong panel, wrong axis, wrong curve, wrong peak, wrong ranking, or wrong comparison target.

Output constraints:
- correct must be a boolean.
- score must be one of 0.0, 0.5, or 1.0.
- error_type must be one of the listed categories.
- explanation must be one short sentence.
- Return JSON only.
```

Figure 8: LLM-as-judge evaluation prompt used only for structured correctness assessment.